Shared Buffer-Based Reverse Scheduling for Onboard Clos-Network Switch

Wanli Chen[1,2], Kai Liu[3], Xiang Chen[1,2(✉)], and Xiangming Kong[4]

[1] School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, Guangdong Province, China
[2] Key Lab of EDA, Research Institute of Tsinghua University in Shenzhen (RITS), Shenzhen 518075, China
chenxiang@mail.sysu.edu.cn
[3] China Academy of Electronics and Information Technology, Beijing 100041, China
[4] Starway Communications Inc., Guangzhou 510663, Guangdong Province, China

Abstract. Onboard switching (OBS) is facing resource constraints and special requirements of hardware complexity and scheduling efficiency. By studying the existing OBS fabrics and scheduling algorithms, the Shared Buffer-based Reverse Scheduling (SB-REV) Algorithm is proposed, adopting the shared buffer in the input module (IM) and guiding the IM scheduling with the matching result of the central modules. Theoretical and experimental analysis shows that the SB-REV algorithm greatly improves the resource utilization and scheduling efficiency, while guaranteeing the cell delay and the throughput performance. The SB-REV Algorithm is highly suitable for resource-constrained OBS environment.

Keywords: Onboard switching · Clos-network · Resource utilization
Scheduling efficiency

1 Introduction

Evolving from 1960s, the satellite communication has gained enormous attentions with the characteristics of large capacity, wide bandwidth, ubiquitous coverage, and the adaptability to multiple services [1]. Compared with the conventional bent-pipe forwarding technology, the onboard switching (OBS) technology only requires end-to-end transmission of one hop, which leads to higher security, lower latency, higher bandwidth utilization and less reliance on ground stations [2]. As the core of the OBS, the OBS fabrics determine the performance of throughput, cell delay, etc. Therefore, the main bottleneck of developing high-speed OBS lies in the OBS fabrics, with many challenges yet to be fulfilled.

In particular, the OBS of China are faced with even worse resource constraints. At present, the payload weight and power of China's largest satellite platform Dongfanghong No. 4 are only 700 kg, 8000 W, whereas that of the

X. Liu et al. (Eds.): ChinaCom 2018, LNICST 262, pp. 719–728, 2019.
https://doi.org/10.1007/978-3-030-06161-6_70

European satellite platform Alphabus are 2000 kg, 18000 W [3], far exceeding China. Since the resources are limited, it is necessary to fully consider the hardware complexity and enhance the resource utilization. On the other hand, as the bandwidth of the OBS gradually increases, it is vital to improve the speed of packet processing and the efficiency of scheduling algorithms. The Reverse Scheduling algorithm was proposed for great enhancement of scheduling efficiency, but with no buffer in the input modules (IM), the scheduling in IM is centralized, decreasing scalability. To solve the problem, crosspoint queue (CQ) can be adopted in IM. But the resource utilization of CQ is low, and thus the structure Shared Buffer was proposed individually, not deployed in the Clos network yet. In order to meet the above challenges, the Shared Buffer-based Reverse Scheduling (SB-REV) Algorithm is proposed, which greatly improves the resource utilization and the scheduling efficiency.

The outline of this paper is as follows. Section 2 presents the central features of the existing OBS fabrics and scheduling algorithms, and then leads to the shared buffer (SB) structure and the reverse scheduling (REV) algorithm. Section 3 analyzes the SB and the REV theoretically. Section 4 shows the performance of the SB and the REV with comprehensive experiment results. The conclusions are drawn in Sect. 5.

2 Related Works

As the switching capacity increases, the OBS fabrics evolve from Time-Division (TD) Switches, suitable for only small-capacity switching, to Space-Division (SD) Switches for large-scale OBS [4]. According to the uniqueness of the switching path, the SD switches can be further divided into single-path switches (such as Crossbar) and multi-path switches (such as Clos) [4]. When the switch size of Crossbar scales, the number of crosspoints added increases exceedingly, while the Clos network free of this problem [5]. The multi-path switch can establish multiple independent paths, and thus the cells of different input-output connections can be forwarded concurrently, with a capacity of up to 10 Tbps. In a Clos network $C(n, m, r)$ (where n, m, r are the number of input links of an input module, central modules and input modules respectively), if $m \geq 2n - 1$, the Clos network is strictly non-breaking [6]. Among multi-path switches, the Clos network is preferable and widely studied for the next-generation OBS, due to its higher reliability, scalability, and the feature of non-blocking.

The Crossbar switch is a key component of the Clos network. According to the existence of buffer inside of it, the Crossbar can be divided into Bufferless Crossbar (including input queue, IQ and output queue, OQ) and Buffered Crossbar (including crosspoint queue, CQ). Because the input and output ports are directly connected, the bufferless Crossbar requires a centralized scheduling algorithm to match the ports. When the switching scales, it becomes difficult for the bufferless Crossbar to meet the requirements of fast scheduling. On the contrary, in the buffered Crossbar (e.g. CQ), the input and output ports are separated. So the concurrent and distributed scheduling algorithm can be implemented, which

reduces the complexity of scheduling and improves the scalability. However, the queue buffer size required for the CQ [7] is proportional to the squared number of ports n^2. To solve this problem, the current solution is adopting the SB [8] (as shown in Fig. 1). Note that each row of crosspoints share the same buffer. Contrasting with the CQ, the required buffer size of the SB is only proportional to the number of ports n, which can be conducive to resource utilizing. Yet the SB has not been deployed in the Clos network in previous literature.

As for the Clos network, it consists of three stages of Crossbar modules, referred to as input modules (IMs), central modules (CMs), and output modules (OMs) respectively. According to the memory deployment of each stage, it can be categorized into SSS-Clos, MSM-Clos, SMM-Clos, and MMM-Clos networks. In the SSS-Clos network, the port matching is actually an $N \times N$ bipartite graph matching problem (in a complexity of $O(N^2)$, where N is the switch size), degrading the scalability [9]. Both the SMM and MMM-Clos networks employ memory at the central module (CM), causing cells out-of-sequence at the receiver. To solve this problem, more buffers and feedbacks are needed, which hinders hardware implementation [10]. Free from the above problems, the MSM-Clos network only needs to solve the conflict at the output port of CM [11]. Herein we will concentrate on the MSM-Clos network.

In the MSM-Clos network, current scheduling algorithms are twofold: dynamic algorithms and quasi-static algorithms. The dynamic algorithm needs to perform real-time routing according to the arriving traffic and employs five-way handshake, which is complicated and time-consuming. The main dynamic algorithm is the Concurrent Round-Robin Dispatching (CRRD) algorithm [12]. In contrast, the quasi-static algorithm pre-configures the connection of CM and cannot adapt to the real-time traffic. To solve the above problems, Zhang et al. [13] proposed the Reverse Scheduling (REV) algorithm, by combining both the dynamic and the quasi-static algorithm. The REV guides the IM scheduling with the matching result of the CM, and cuts down the handshake times and the scheduling time. However in the IM, it's still using the bufferless Crossbar aforementioned, which needs centralized scheduling and lacks scalability.

In this article, we will combine the advantages of both the shared buffer and the reverse scheduling, propose the SB-REV algorithm, and prove its superiority by theoretical and experimental analysis.

3 The Shared Buffer-Based Reverse Scheduling Algorithm

3.1 The Shared Buffer Fabrics (SB)

In the crosspoint queue CQ_{ij}, only the cells from the input port I_i to the output port O_j can be buffered. By comparison, in the shared buffer SB_i as shown in Fig. 1, cells from the input port I_i to any output port $\mathrm{O}_j(\forall j)$ can be buffered.

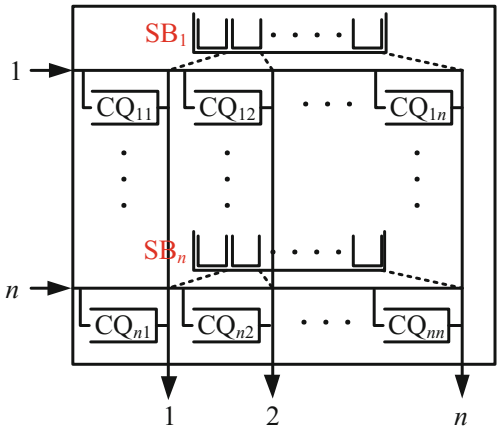

Fig. 1. The shared buffer fabrics

Accordingly we have the relationship between the queue length of the SB and the CQ $L_{SB} = sum(L_{CQ})$, where $L_{CQ}, L_{SB} \in [0, n]$. The required buffer size, denoted by L_{set}, is set according to the maximum queue length L_{max} when the queue length is not limited in the experiment. Consequently, we have

$$\frac{L_{set\_SB}}{L_{set\_CQ}} = \frac{n \times L_{max\_SB}}{n^2 \times L_{max\_CQ}} \in \left[\frac{1}{n}, 1\right], \tag{1}$$

where L_{max_SB} and L_{max_CQ} are the maximum queue lengths among n SBs and among n^2 CQs respectively. In other words, the required buffer size of the SB can be reduced up to $1/n$ that of the CQ. Note that the buffer utilization r is the ratio between the actually used buffer size and the required buffer size, i.e., $r = L_{act}/L_{set}$. If the SB and the CQ have the same L_{act} in the experiment, by Formula (1) we have

$$\frac{r_{SB}}{r_{CQ}} = \frac{\frac{L_{act\_SB}}{L_{set\_SB}}}{\frac{L_{act\_CQ}}{L_{set\_CQ}}} \in [1, n], \tag{2}$$

i.e., the buffer utilization of the SB can be improved up to n times that of the CQ.

3.2 The Reverse Scheduling Algorithm (REV)

In the MSM-Clos network, the concurrent round-robin dispatching (CRRD) algorithm and the reverse scheduling (REV) algorithm are compared.

(1) Hardware complexity
The core component of the scheduling is the arbitration scheduler, so the hardware complexity of the scheduler represents the resource overhead.

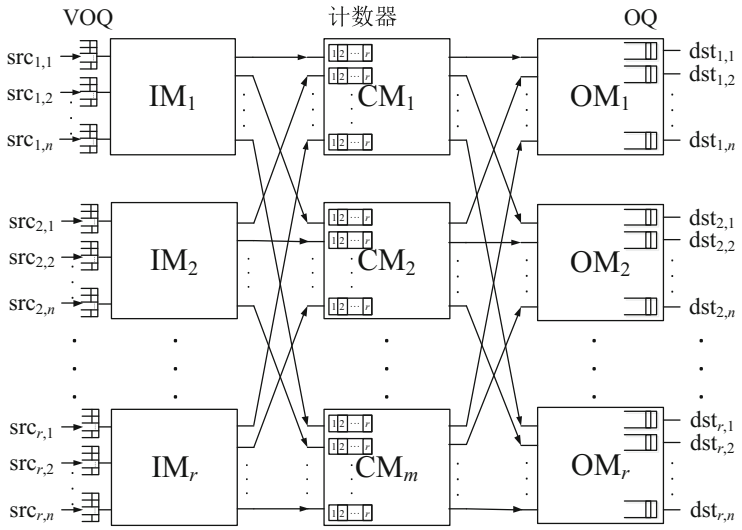

Fig. 2. The reverse scheduling algorithm in the MSM-Clos network

When adopting the CRRD, the hardware complexity of the r IMs is $O(nmr)$, so the complete hardware complexity is $O(mrN)$ [13]. When adopting the REV, as shown in Fig. 2, only the input port CI_{ji} in the CM needs a scheduler, so the hardware complexity of the m CMs is $O(r^2)$, and the complete hardware complexity is $O(mr^2)$. Consequently, the ratio of hardware complexity between the two algorithms is $\frac{O(\mathrm{REV})}{O(\mathrm{CRRD})} = \frac{O(mr^2)}{O(mrN)} = \frac{O(r)}{O(N)} = \frac{1}{n}$. That is, the resource overhead of the REV is much lower than that of the CRRD, which is conducive to solving the resource constraints of the OBS.

(2) Scheduling time

The scheduler is responsible for the decision of the handshakes or the matchings between the input and output ports. For a scheduler with n input numbers, the time complexity of the scheduler is $O(logn)$ [13]. When adopting the REV, the scheduling is only required in the CM. Assume that the Round-Robin (RR) algorithm is adopted in the CM to avoid starvation, then the scheduling time of the REV is proportional to $logr$, i.e., $t_{arb}(\mathrm{REV}) = \alpha logr$, where α is a constant coefficient, determined by actual hardware performance.

As for the CRRD, it requires five-way handshake. Let $i(i \geq 1)$ be the iteration times in the IM, then the scheduling time of the CRRD is $t_{arb}(\mathrm{CRRD}) = \alpha[i(logN + logm) + logr]$. So when $n = m = r$, we have [13]

$$\frac{t_{arb}(\mathrm{REV})}{t_{arb}(\mathrm{CRRD})} = \frac{1}{1 + ilog_r nmr} \overset{n=m=r}{=} \frac{1}{1+3i} = \frac{1}{4}. \tag{3}$$

4 Simulation Analysis

4.1 The Shared Buffer Fabrics (SB)

This paper uses the software OPNET to build the environment of the OBS simulation. In the experiment, we adopt the RR algorithm, simulate for 10 000 timeslots in a 16×16 Crossbar ($n = 16$), and compare the SB with the CQ. Here we define symbols $\eta_{L_{act}}, \eta_{L_{set}}, \eta_r$ to represent the ratios between the SB and the CQ in terms of the actually used buffer size, the required buffer size and the buffer utilization. Specifically, the actually used buffer size is proportional to the average of the actually used buffer size in the experiment $L_{act} \propto L_{avg}$, and the required buffer size is proportional to the maximum used buffer size $L_{set} \propto L_{max}$.

Under Bernoulli traffic, adopting the SB or the CQ can both achieve 100% throughput, but they differs in terms of the required buffer size. Experimental results are shown in Fig. 3. When the Bernoulli traffic load $\lambda \leq 0.5$, we have $\eta_{L_{set}} = \eta_{L_{max}} = L_{max\_SB}/L_{max\_CQ} = 1/16 = 1/n$, and $\eta_{L_{act}} = \eta_{L_{avg}} = L_{avg\_SB}/L_{avg\_CQ} = 1$. So the ratio of the buffer utilization between the SB and the CQ is $\eta_r = \eta_{L_{act}}/\eta_{L_{set}} = 16 = n$.

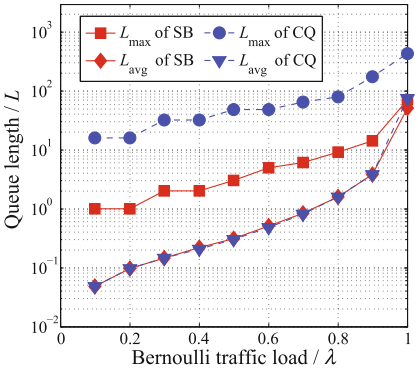

Fig. 3. Queue length of CQ and SB

Table 1. The throughput and cell delay of the iterative CRRD and REV algorithm

CRRD	Iterations	Cell delay	ρ_1	ρ_2
	1	389.44	0.99214	0.90800
	2	41.39	0.99982	0.97915
	3	40.80	1	0.97738
	4	40.80	1	0.97738
	5	40.80	1	0.97738
REV		46.24	N/A	0.99827

4.2 The Reverse Scheduling Algorithm (REV)

In the experiment, we simulate for 10 000 timeslots under the Bernoulli traffic ($\lambda = 1$) in the Clos network $C(8,8,8)$, and compare the REV with the CRRD$_i$ ($i = 1, 2, \ldots, 5$). Here we define the matching rate between the input and output

ports in the IM as ρ_1, and the matching rate in the CM as ρ_2. Consequently we have the throughput of IM $\rho_{\text{IM}} = \rho_1 \times \rho_2$.

Experimental results are shown in Table 1. At the iterations of $i = 1, 2, 3$, we have the matching rate in the IM $\rho_1 = 99.214\%, 99.982\%, 100\%$, respectively. Iteration times higher than three cannot contribute to the matching rate and the cell delay, but further decreases the actual scheduling efficiency. On the contrary, the REV need no iteration to achieve a throughput of 99.8%, higher than that of the $\text{CRRD}_i(\forall i)$.

4.3 The Shared Buffer-Based Reverse Scheduling Algorithm

In the experiment, we simulate for 10 000 timeslots under the Bernoulli traffic ($\lambda = 1$) in the Clos network C(8,8,8), and compare seven algorithms. In order to avoid the Head-of-Line (HoL) Blocking, the virtual output queue (VOQ) is adopted [14] in input stages. Here, the scheduling algorithms involving the CRRD iterate once.

According to the queuing strategy in the IM, seven algorithms can be divided into the following three groups:

(1) Bufferless Crossbar: the CRRD and the REV without buffer inside of the Crossbar;
(2) SB: the CRRD based on the SB and that with speedup (SB-CRRD and SB-CRRD_S) and the REV based on the SB and that with speedup (SB-REV and SB-REV_S);
(3) CQ: the CRRD based on the CQ (CQ-CRRD).

Among them, the second and the third groups of algorithms are based on Buffered Crossbar, which is conducive to concurrent and distributed scheduling and thus enhances scalability.

Specifically, there are two mechanisms of the RR scheduling in the SB: (1) The polling objects are all $n \times m$ cells in the buffer; (2) The polling objects are $n \times 1$ shared buffers (where n, m are the number of input and output ports of the IM respectively). The effect of adopting mechanism (1) is completely equivalent to that of not employing the SB, and thus mechanism (1) is not discussed specifically. When adopting mechanism (2), the polling number decreases to n, and the polling efficiency increases to m times of that without the SB. In this paper, when some shared buffer is successfully authorized, we only allow the oldest cell in it to be served first. Hereafter we adopt mechanism (2) in the SB by default.

As shown in Fig. 4a, the throughput of algorithms except CRRD, SB-CRRD, and SB-REV, reaches 100%. However, the throughput of SB-CRRD is even lower than that of CRRD. This is because we adopt mechanism (2) in the SB aforementioned, which causes the non-oldest cell in the buffer losing the right to poll, but meanwhile increases the RR scheduling efficiency up to m times. Different from the CQ, the cells in the SB can go to any output port. So as shown in Fig. 6, if some SB (SB_1) is not empty and some output port (O_2) is idle, then the throughput can be improved with a speedup. As shown in Fig. 7, the CQ is

limited by the fixed output link rate and cannot increase the throughput with a speedup.

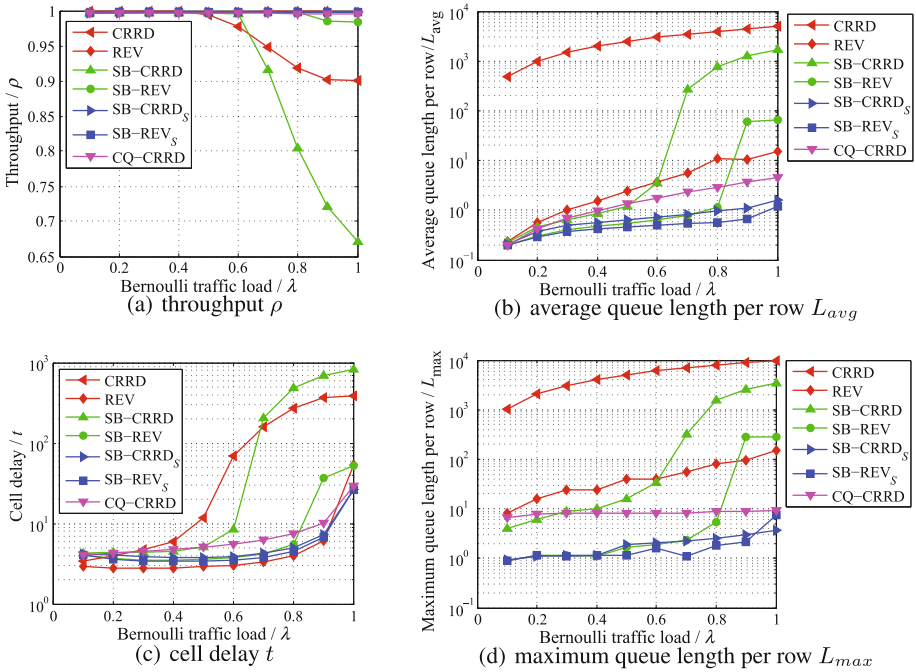

Fig. 4. Comparison of 7 algorithms in 4 different performances

As shown in Fig. 4a, SB-CRRD can achieve 100% throughput with a speedup, and as shown in Fig. 5, the speedup of SB-CRRD converges to $S = 1.208$. Similarly, the SB-REV algorithm has a throughput of 98.4% without speedup. And as shown in Fig. 5, the SB-REV needs a speedup of only $S = 1.05$ to outperform other algorithms in terms of throughput, cell delay and queue overhead.

As shown in Fig. 4a, c, the CQ-CRRD throughput is 100% and the cell delay performance is optimal. However, it can be seen from Fig. 4b, d that the queue overhead of the CQ-CRRD is large. Calculations show that the average required buffer size $\overline{L}_{set} = \frac{1}{10}\sum_{\lambda=0.1}^{1} L_{max}(\lambda)$ of the CQ-CRRD is 4 times that of the SB-REV$_S$, much inferior to the SB-REV$_S$.

As can be seen from the four aspects in Fig. 4a, b, c, d, using the SB does not necessarily optimize the performance. However compared with the CRRD, adopting the REV can optimize the performance of all aspects. Among the seven algorithms, the SB-REV$_S$ outperforms others remarkably.

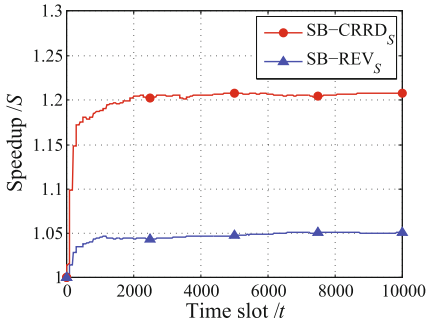

Fig. 5. Speedup performance when Bernoulli traffic load $\lambda = 1$

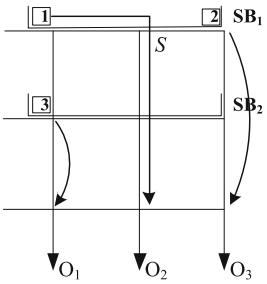

Fig. 6. SB improves throughput with speedup

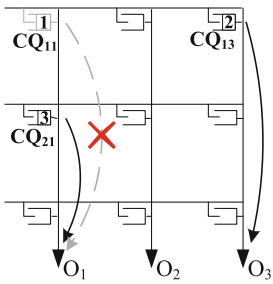

Fig. 7. CQ cannot speed up

5 Conclusions

This paper focuses on the problem of resource constraints of the OBS. Starting from the resource utilization and the scheduling efficiency, we study the shared buffer structure and the reverse scheduling algorithm, and propose the SB-REV algorithm. Theoretical analysis shows that the queue buffer size is reduced up to $1/n$, that the hardware complexity is reduced to $O(1/n)$, and that the scheduling time is reduced to $1/(1 + 3i)$ (where i is the iteration times of the compared algorithm CRRD).

This paper uses the software OPNET to build the environment of the OBS simulation. Experiment results show that: (1) By adopting the SB, the memory utilization is increased to n times that of the CQ; (2) By adopting the REV, no iteration is required, yielding a throughput ($\rho = 99.8\%$) higher than that of the CRRD$_i(\forall i)$. In addition, we also figure out the effects of adopting the SB and/or the REV from different points. Simulations show that the SB-REV algorithm only needs a speedup of $S \leq 1.05$ to outperform other algorithms in terms of cell delay and throughput. With lower hardware complexity and higher scheduling efficiency, the SB-REV algorithm is suitable for the resource-constrained onboard switching.

Acknowledgement. The work is supported by the NSFC (No. 61501527), Science, Technology and Innovation Commission of Shenzhen Municipality (No. JCYJ20170816151823313), Foundation for Innovation by China Academy of Electronics and Information Technology "Research on Protocol Oblivious Forwarding Technologies for Space Network", Guangdong Innovative and Entrepreneurial Research Team Program (No. 2013D014), China's Postdoctoral Science Foundation (No. 2017M620061), State's Key Project of Research and Development Plan (No. 2016YFE0122900-3), the Fundamental Research Funds for the Central Universities and 2016 Major Project of Collaborative Innovation in Guangzhou (No. 201604046008).

References

1. He, Y.Z.: Research on new genergation mobile satellite communication system[D]. Beijing University of Posts and Telecommunications (2015)
2. Wang, J., Qiao, L., Shao, S., et al.: High-performance routing search algorithm in satellite IP switches[C]. In: Proceedings of IEEE Computer Science and Network Technology, pp. 863–866. Dalian (2013)
3. Wang, M., Zhou, Z.C.: Analysis of the alphabus platform devel and design characteristics[J]. Spacecr. Eng. **19**(2), 99–105 (2010)
4. Chao, H.J., Liu, B.: High Performance Switches and Routers. Wiley, New York (2007)
5. Yang, W.X., et al.: Design and implementation of a multi-stage bufferless high radix router[J]. Comput. Eng. Sci. **39**(2), 245–251 (2017)
6. Tang, H.K.: Load balance technology of the Clos network[J]. Sci. Technol. Inf. **15**(8), 7–9 (2017)
7. Kleban, J., Suszynska, U.: Static dispatching with internal backpressure scheme for SMM Clos-network switches[C]. In: Computers and Communications, pp. 000654–000658. IEEE (2014)
8. Kornaros, G.: BCB: A Buffered CrossBar switch fabric utilizing shared memory[C]. In: Euromicro Conference on Digital System Design, pp. 180–188. IEEE Computer Society (2006)
9. Chao, H.J., Jing, Z., Liew, S.Y., et al.: Matching algorithms for three-stage bufferless Clos network switches[J]. IEEE Commun. Mag. **10**, 46–54 (2003)
10. Dong, Z., Rojas-Cessa, R., Oki, E.: Memory-memory-memory Clos-network packet switches with in-sequence service[C]. In: IEEE, International Conference on High PERFORMANCE Switching and Routing, pp. 121–125. IEEE (2011)
11. Gao, Y., Qiu, Z., Zhang, M., et al.: Distributed weight matching dispatching scheme in MSM Clos-network packet switches[J]. IEEE Commun. Lett. **17**(3), 580–583 (2013)
12. Oki, E., Jing, Z., Rojas-Cessa, R., et al.: Concurrent round-robin-based dispatching schemes for Clos-network switches. IEEE/ACM Trans. Netw. **10**(6), 830–844 (2002)
13. Zhang, M., Qiu, Z., Gao, Y., et al.: Reverse dispatching scheme for satellite Clos-network switches[J]. J. Xidian Univ. **40**(4), 96–101 (2013)
14. Kleban, J.: Packet dispatching using module matching in the modified MSM Clos-network switch[J]. Telecommun. Syst. **8**, 1–9 (2017)